Attorney Docket No.: 6502.0372

Express Mail Label No.: EL423323606US

UNITED STATES PATENT APPLICATION

FOR

FLOATING POINT STATUS INFORMATION TESTING CIRCUIT

BY

GUY L. STEELE, JR.

RELATED APPLICATIONS

[001] U.S. Patent Application Serial No. _____, filed on even date herewith in the name of Guy L. Steele Jr. and entitled "Floating Point Unit in which Floating Point Status Information is Encoded in Floating Point Representations," assigned to the assignee of the present application, is hereby incorporated by reference.

DESCRIPTION OF THE INVENTION

Field of the Invention

[002] The invention related generally to systems and methods for performing floating point operations, and more particularly to systems and methods for selectively testing a floating point operand determining the condition of floating point status information associated with a floating point operand.

Background of the Invention

[003] Digital electronic devices, such as digital computers, calculators and other devices, perform arithmetic calculations on values in integer, or "fixed point," format, in fractional, or "floating point" format, or both. Institute of Electrical and Electronic Engineers (IEEE) Standard 754, (hereinafter "IEEE Std. 754" or "the Standard") published in 1985 and adopted by the American National Standards Institute (ANSI), defines several standard formats for expressing values in floating point format, and a number of aspects regarding behavior of computation in connection therewith. In accordance with IEEE Std. 754, a representation in floating point format comprises a plurality of binary digits, or "bits," having the structure

[004] $se_{msb}\cdots e_{lsb}f_{msb}\cdots f_{lsb}$

where bit "s" is a sign bit indicating whether the entire value is positive or negative, bits "$e_{msb}\cdots e_{lsb}$" comprise an exponent field that represents the exponent "e" in unsigned binary biased format, and bits "$f_{msb}\cdots f_{lsb}$" comprise a fraction field that represents the fractional portion "f" in unsigned binary format ("msb" represents "most significant bit" and "lsb" represents "least significant bit"). The Standard defines two general formats. A "single" format comprises thirty-two bits while a "double" format comprises sixty-four bits. In the single format, there is one sign bit "s," eight bits "$e_7\ldots e_0$" comprising the exponent field and twenty-three bits "$f_{22}\ldots f_0$" comprising the fraction field. In the double format, there is one sign bit "s," eleven bits "$e_{10}\ldots e_0$" comprising the exponent field and fifty-two bits "$f_{51}\ldots f_0$" comprising the fraction field.

[005] As indicated above, the exponent field of the floating point representation "$e_{msb}\cdots e_{lsb}$" represents the exponent "E" in biased format. The biased format provides a mechanism by which the sign of the exponent is implicitly indicated. In particular, the bits "$e_{msb}\cdots e_{lsb}$" represent a binary encoded value "e" such that "e=E + bias." This allows the exponent E to extend from -126 to +127, in the eight-bit "single" format, and from -1022 to +1023 in the eleven-bit "double" format, and provides for relatively easy manipulation of the exponents in multiplication and division operations, in which the exponents are added and subtracted, respectively.

[006] IEEE Std. 754 provides for several different formats with both the single and double formats which are generally based on the bit patterns of the bits

"$e_{msb}\cdots e_{lsb}$" comprising the exponent field and the bits "$f_{msb}\cdots f_{lsb}$" comprising the fraction field. If a number is represented such that all of the bits "$e_{msb}\cdots e_{lsb}$" of the exponent field are binary one's (*i.e.*, if the bits represent a binary-encoded value of "255" in the single format or "2047" in the double format) and all of the bits "$f_{msb}\cdots f_{lsb}$" of the fraction field are binary zeros, then the value of the number is positive or negative infinity, depending on the value of the sign bit "s." In particular, the value "v" is $v = (-1)^s \infty$, where "∞" represents the value "infinity." On the other hand, if all of the bits "$e_{msb}\cdots e_{lsb}$" of the exponent field are binary one's and if the bits "$f_{msb}\cdots f_{lsb}$" of the fraction field are not all zero's, then the value that is represented is deemed "not a number," which is abbreviated in the Standard by "NaN."

[007] If a number has an exponent field in which the bits "$e_{msb}\cdots e_{lsb}$" are neither all binary ones nor all binary zeros (*i.e.*, if the bits represent a binary-encoded value between 1 and 254 in the single format or between 1 and 2046 in the double format), the number is said to be a "normalized" format. For a number in the normalized format, the value represented by the number is $v = (-1)^s \; 2^{e\text{-bias}}(1.f_{msb}\cdots f\cdots_{lsb})$, where "." represents a concatenation operation. Effectively, in the normalized format, there is an implicit most significant digit having the value "one," so that the twenty-three digits in the fraction field of the single format, or the fifty-two digits in the fraction field of the double format, will effectively represent a value having twenty-four digits or fifty-three digits of precision, respectively, where the value is less than two, but not less than one.

[008] On the other hand, if a number has an exponent field in which the bits "$e_{msb}\cdots e_{lsb}$" are all binary zeros, representing the binary-encoded value of "zero," and

a fraction field in which the bits $f_{msb} \cdots f_{lsb}$ are not all zero, the number is said to be a "de-normalized" format. For a number in the de-normalized format, the value represented by the number is $v = (-1)^s 2^{e-bias+1}(0.f_{msb} \cdots f_{lsb})$. It will be appreciated that the range of values of numbers that can be expressed in the de-normalized format is disjoint from the range of values of numbers that can be expressed in the normalized format, for both the single and double formats. Finally, if a number has an exponent field in which the bits "$e_{msb} \cdots e_{lsb}$" are all binary zeros, representing the binary-encoded value of "zero," and a fraction field in which the bits $f_{msb} \cdots f_{lsb}$ are all zero, the number has the value "zero". It will be appreciated that the value "zero" may be positive zero or negative zero, depending on the value of the sign bit.

[009] Generally, circuits or devices that perform floating point computations or operations (generally referred to as floating point units) conforming to IEEE Std. 754 are designed to generate a result in three steps:

[010] (a) In the first step, an approximation calculation step, an approximation to the absolutely accurate mathematical result (assuming that the input operands represent the specific mathematical values as described by IEEE Std. 754) is calculated that is sufficiently precise as to allow this accurate mathematical result to be summarized. The summarized result is usually represented by a sign bit, an exponent (typically represented using more bits than are used for an exponent in the standard floating-point format), and some number "N" of bits of the presumed result fraction, plus a guard bit and a sticky bit. The value of the exponent will be such that the value of the fraction generated in step (a) consists of a 1 before the binary point and a fraction after the binary point. The bits

are commonly calculated so as to obtain the same result as the following conceptual procedure (which is impossible under some circumstances to carry out in practice): calculate the mathematical result to an infinite number of bits of precision in binary scientific notation, and in such a way that there is no bit position in the significand such that all bits of lesser significance are 1-bits (this restriction avoids the ambiguity between, for example, 1.100000... and 1.011111... as representations of the value "one-and-one-half"); let the N most significant bits of the infinite significand be used as the intermediate result significand; let the next bit of the infinite significand be the guard bit; and let the sticky bit be 0 if and only if ALL remaining bits of the infinite significant are 0-bits (in other words, the sticky bit is the logical OR of all remaining bits of the infinite fraction after the guard bit).

[011] (b) In the second step, a rounding step, the guard bit, the sticky bit, perhaps the sign bit, and perhaps some of the bits of the presumed significand generated in step (a) are used to decide whether to alter the result of step (a). For conventional rounding modes defined by IEEE Std. 754, this is a decision as to whether to increase the magnitude of the number represented by the presumed exponent and fraction generated in step (a). Increasing the magnitude of the number is done by adding 1 to the significand in its least significant bit position, as if the significand were a binary integer. It will be appreciated that, if the significand is all 1-bits, the magnitude of the number is "increased" by changing it to a high-order 1-bit followed by all 0-bits and adding 1 to the exponent.

[012] Regarding the rounding modes, it will be further appreciated that,

[013] (i) if the result is a positive number, and

[014] (a) if the decision is made to increase, effectively the decision has been made to increase the value of the result, thereby rounding the result up (*i.e.,* towards positive infinity), but

[015] (b) if the decision is made not to increase, effectively the decision has been made to decrease the value of the result, thereby rounding the result down (*i.e.,* towards negative infinity); and

[016] (ii) if the result is a negative number, and

[017] (a) if the decision is made to increase, effectively the decision has been made to decrease the value of the result, thereby rounding the result down, but

[018] (b) if the decision is made not to increase, effectively the decision has been made to increase the value of the result, thereby rounding the result up.

[019] (c) In the third step, a packaging step, a result is packaged into a standard floating-point format. This may involve substituting a special representation, such as the representation defined for infinity or NaN if an exceptional situation (such as overflow, underflow, or an invalid operation) was detected. Alternatively, this may involve removing the leading 1-bit (if any) of the fraction, because such leading 1-bits are implicit in the standard format. As another alternative, this may involve shifting the fraction in order to construct a denormalized number. As a specific example, it is assumed that this is the step that forces the result to be a NaN if any input operand is a NaN. In this step, the decision is also made as to whether the result should be an infinity. It will be appreciated that, if the

result is to be a NaN or infinity, the original result will be discarded and an appropriate representation will be provided as the result.

[020] In addition in the packaging step, floating point status information is generated, which is stored in a floating point status register. The floating point status information generated for a particular floating point operation includes indications, for example, as to whether

[021] (i) a particular operand is invalid for the operation to be performed ("invalid operation");

[022] (ii) if the operation to be performed is division, the divisor is zero ("division-by-zero");

[023] (iii) an overflow occurred during the operation ("overflow");

[024] (iv) an underflow occurred during the operation ("underflow"); and

[025] (v) the round result of the operation is not exact ("inexact").

[026] These conditions are typically represented by flags that are stored in the floating point status register. The floating point status information can be used to dynamically control the operations in response to certain instructions, such as conditional branch, conditional move, and conditional trap instructions that may be in the instruction stream subsequent to the floating point instruction. Also, the floating point status information may enable processing of a trap sequence, which will interrupt the normal flow of program execution. In addition, the floating point status information may be used to affect certain ones of the functional unit control signals that control the rounding mode. IEEE Std. 754 also provides for accumulating

floating point status information from, for example, results generated for a series or plurality of floating point operations.

[027] IEEE Std. 754 has brought relative harmony and stability to floating-point computation and architectural design of floating-point units. Moreover, its design was based on some important principles, and rests on a sensible mathematical semantics that eases the job of programmers and numerical analysts. It also supports the implementation of interval arithmetic, which may prove to be preferable to simple scalar arithmetic for many tasks. Nevertheless, IEEE Std. 754 has some serious drawbacks, including:

[028] (i) Modes (*e.g.*, the rounding modes and traps enabled/disabled mode), flags (*e.g.*, flags representing the status information and traps required to implement IEEE Std. 754 introduce implicit serialization issues. Implicit serialization is essentially the need for serial control of access (read/write) to and from globally used registers, such as a conventional floating point status register. Under IEEE Std. 754, implicit serialization may arise between (1) different concurrent floating-point instructions and (2) between floating point instructions and the instructions that read and write the flags and modes. Furthermore, rounding modes may introduce implicit serialization because they are typically indicated as global state, although in some microprocessor architectures, the rounding mode is encoded as part of the instruction operation code, which will alleviate this problem to that extent. Thus, the potential for implicit serialization makes the Standard difficult to implement coherently and efficiently in today's superscalar and parallel processing architectures without loss of performance.

[029] (ii) The implicit side effects of a procedure that can change the flags or modes can make it very difficult for compilers to perform optimizations on floating point code. As a result, compilers for most languages usually assume that every procedure call is an optimization barrier in order to be safe. This unfortunately may lead to further loss of performance.

[030] (iii) Global flags, such as those that signal certain modes, make it more difficult to do instruction scheduling where the best performance is provided by interleaving instructions of unrelated computations. Thus, instructions from regions of code governed by different flag settings or different flag detection requirements cannot easily be interleaved when they must share a single set of global flag bits.

[031] (iv) Furthermore, traps have been difficult to integrate efficiently into computing architectures and programming language designs for fine-grained control of algorithmic behavior.

[032] Thus, there is a need for a system that avoids such problems when performing floating point operations and, in particular, when testing for status information related to a floating point operand.

SUMMARY OF THE INVENTION

[033] Methods, systems, and articles of manufacture consistent with the present invention overcome these shortcomings with a floating point testing circuit for testing a floating point operand having status information encoded within the operand itself. This circuit determines the status condition of the operand without requiring use or access to a separate floating point status register and identifies whether the status condition is of a selected type.

[034] More particularly stated, a floating point operand testing circuit consistent with the present invention, as embodied and broadly described herein, includes an analysis circuit and a result generator circuit coupled to the analysis circuit. The analysis circuit can determine the status of the floating point operand based upon data within the floating point operand. An operand buffer may be coupled to the analysis circuit and configured to supply the floating point operand to the analysis circuit.

[035] The result generator circuit is responsive to at least one control signal. Further, the result generator is configured to assert one or more result signals if the floating point analysis circuit indicates the floating point status is of a predetermined format specified by the at least one control signal. The result signal may be used to condition the outcome of a floating point instruction.

[036] The result generator circuit may also be responsive to at least one of a plurality of control signals that is asserted when testing the same floating point operand for one of a plurality of predetermined formats.

[037] The data within the floating point operand typically encodes the status in the predetermined format. Such a predetermined format may be one or a combination from a group comprising not-a-number (NaN), infinity, normalized, denormalized, invalid operation, overflow, underflow, division by zero, exact, and inexact. In more detail, the predetermined format may represent a +OV status, a -OV status, a +UN status, a -UN status, a positive infinity status or a negative infinity status. The predetermined format may also comprise a plurality of bits indicative of

a predetermined type of operand condition resulting in the NaN status or the infinity status.

[038] The result generator may also update the result signal to indicate that the floating point operand is of an alternative format specified by an updated value of the at least one control signal. Similar to the predetermined format, the alternative format may be one or a combination of NaN, infinity, normalized, denormalized, invalid operation, overflow, underflow, division by zero, exact, or inexact.

[039] In another aspect of the present invention, a floating point operand testing circuit consistent with an embodiment of the present invention is described that includes an analysis circuit, an operand buffer and a result generator circuit. The analysis circuit is configured to determine the status of the floating point operand based only on the contents of the floating point operand. The operand buffer is coupled to the analysis circuit and configured to store the floating point operand encoded with the status.

[040] The result generator circuit is coupled to the analysis circuit and provides a result signal in response to at least one control signal. The result signal is indicative of whether the status of the floating point operand conforms to a predetermined format associated with the control signal. The control signal may be implemented as a group of control signals that are asserted in order to test the floating point operand for different status conditions.

[041] The predetermined format may be one or a combination from a group comprising NaN, infinity, normalized, denormalized, invalid operation, overflow, underflow, division by zero, exact, or inexact. In more detail, the predetermined

format may represent a +OV status, a -OV status, a +UN status, a -UN status, a positive infinity status and a negative infinity status. Further, the predetermined format may include a bits indicative of a predetermined type of operand condition resulting in the NaN status.

[042] In yet another aspect of the present invention, a method for testing a floating point status condition of a floating point operand consistent with an embodiment of the present invention is described that begins with receiving the floating point operand. The floating point status of the operand is determined from only the contents of the floating point operand. At least one control signal is received and then a result signal is generated. The result signal indicates whether the status of the floating point operand conforms to a predetermined format associated with the control signal. The stages of receiving a control signal and generating a result signal may be repeated in order to test the same operand for different status conditions.

[043] The predetermined format may include one or a combination of not-a-number (NaN), infinity, normalized, denormalized, invalid operation, overflow, underflow, division by zero, exact, or inexact. In more detail, the predetermined format may represent a +OV status, a -OV status, a +UN status, a -UN status, a positive infinity status or a negative infinity status. Furthermore, the predetermined format for the NaN may include bits indicative of a predetermined type of operand condition resulting in the NaN status.

[044] Additionally, the method may include conditioning the outcome of a floating point instruction based upon the value of the result signal. This

advantageously allows for conditional processing without relying upon the contents of a separate floating point status register, which may have undesirably been over-written.

[045] In still another aspect of the present invention, a computer-readable medium is described that is consistent with an embodiment of the present invention. The medium stores a set of instructions for testing a floating point status condition of a floating point operand. When executed, the instructions perform a method as described above or herein.

[046] Additional advantages of the invention will be set forth in part in the description which follows, and in part will be obvious from the description, or may be learned by practice of the invention. The advantages of the invention will be realized and attained by means of the elements and combinations particularly pointed out in the appended claims.

[047] It is to be understood that both the foregoing general description and the following detailed description are exemplary and explanatory only and are not restrictive of the invention, as claimed.

[048] The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate embodiments of the invention and together with the description, serve to explain the principles of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[049] Figure 1 is a functional block diagram of an exemplary tester unit for selectively determining the condition of status information associated with a floating

point operand, consistent with an exemplary embodiment of the present invention; and

[050] Figure 2 depicts exemplary formats for representations of floating point operands, the status of which is determined by the exemplary tester unit depicted in Figure 1 and which is consistent with an exemplary embodiment of the present invention.

DESCRIPTION OF THE EMBODIMENTS

[051] Reference will now be made in detail to exemplary embodiments of the present invention, examples of which are illustrated in the accompanying drawings. Wherever possible, the same reference numbers will be used throughout the drawings to refer to the same or like parts.

[052] Related U.S. Patent Application Serial No. _____, which has previously been incorporated by reference, describes an exemplary floating point unit in which floating point status information is encoded in the representations of the results generated thereby. The exemplary floating point unit includes a plurality of functional units, including an adder unit, a multiplier unit, a divider unit, a square root unit, a maximum/minimum unit, a comparator unit and a tester unit, all of which operate under control of functional unit control signals provided by a control unit. The present application is directed to an exemplary tester unit that can be used in floating point operations with the floating point unit described in related U.S. Patent Application Serial No. _____.

[053] Figure 1 is a functional block diagram of an exemplary tester unit 10 for selectively determining the condition of status information associated with a

floating point operand, constructed in accordance with the invention. Generally, the tester unit 10 receives a floating point operand and generates a result selectively indicating the condition of status information that is encoded therein and comprises a part thereof. Since the floating point status information comprises part of the floating point representation of the operand instead of being separate and apart from the operand as in prior art tester units, there is no need nor desire to access any external circuitry (*e.g.*, floating point status register). Thus, the implicit serialization that is required by maintaining the floating point status information separate and apart from the operand can be advantageously obviated.

[054] The operands received by the tester circuit 10 can be in ones of a plurality of exemplary formats, which will be illustrated in connection with Figure 2. With reference to Figure 2, seven exemplary formats are depicted for floating point operands tested by tester circuit 10. These exemplary formats include a zero format 100, an underflow format 101, de-normalized format 102, a normalized non-zero format 103, an overflow format 104, an infinity format 105 and a not-a-number (NaN) format 106. The exemplary zero format 100 is used to represent the values "zero," or, more specifically, positive or negative zero, depending on the value of "s," the sign bit.

[055] The exemplary underflow format 101 provides a mechanism by which the floating point status information testing circuit 10 can determine that the result of a computation is an overflow. In the underflow format, the sign bit "s" indicates whether the result is positive or negative, the bits $e_{msb} \cdots e_{lsb}$ of the exponent field are all binary zero's, and the bits $f_{msb} \cdots f_{lsb+1}$ of the fraction field, except for the least

significant bit, are all binary zero's. The least significant bit f_{lsb} of the fraction field is a binary one.

[056] The exemplary de-normalized format 102 and exemplary normalized non-zero format 103 are used to represent finite non-zero floating point values substantially along the lines of that described above in connections with IEEE Std. 754. In both formats 102 and 103, the sign bit "s" indicates whether the result is positive or negative. The bits $e_{msb}\cdots e_{lsb}$ of the exponent field of the de-normalized format 102 are all binary zero's. However, the bits $e_{msb}\cdots e_{lsb}$ of the exponent field of the normalized non-zero format 103 are mixed one's and zero's, except that the exponent field of the normalized non-zero format 103 will not have a pattern in which bits $f_{msb}\cdots f_{lsb+1}$ are all binary ones and the least significant bit e_{lab} zero and all of the bits $f_{msb}\cdots f_{lsb}$ of the fraction field are all binary one's. In exemplary format 102 the bits $f_{msb}\cdots f_{lsb+1}$ of the fraction field are not all binary zero's.

[057] The exemplary overflow format 104 provides a mechanism by which the floating point status information testing circuit 10 can determine that the result of a computation is an overflow. In the overflow format 104, the sign bit "s" indicates whether the result is positive or negative, the bits $e_{msb}\cdots e_{lsb+1}$ of the exponent field are all binary ones, with the least significant bit e_{lsb} being zero. The bits $f_{msb}\cdots f_{lsb}$ of the fraction field are all binary ones.

[058] The exemplary infinity format 105 provides a mechanism by which the floating point status information testing circuit 10 can determine that its operand is infinite. In the infinity format 105, the sign bit "s" indicates whether the result is positive or negative, the bits $e_{msb}\cdots e_{lsb}$ of the exponent field are all binary ones, and

the bits $f_{msb}\cdots f_{lsb+5}$ of the fraction field are all binary zero's. The five least significant bits $f_{lsb+4}\cdots f_{lsb}$ of the fraction field are flags, which will be further described below.

[059] The exemplary NaN (not-a-number) format 106 provides a mechanism by which the floating point status information testing circuit 10 can determine that a result is not a number. In the NaN format, the sign bit "s" can be any value, the bits $e_{msb}\cdots e_{lsb}$ of the exponent field are all binary ones, and the bits $f_{msb}\cdots f_{lsb+5}$ of the fraction field are not all binary zero's. The five least significant bits $f_{lsb+4}\cdots f_{lsb}$ of the fraction field are flags, which will be further described below.

[060] As noted above, in values represented in the exemplary infinity format 105 and the exemplary NaN format 106, the five low order bits $f_{lsb+4}\cdots f_{lsb}$ of the fraction field are flags. In one embodiment, such as the exemplary formats used with the floating point status information testing circuit 10, the five flags include the flags that are defined by IEEE Std. 754. These flags include an invalid operation flag "n," an overflow flag "o," an underflow flag "u," a division-by-zero flag "z," and an inexact flag "x." For example, a value in the exemplary NaN format 106 in which both the overflow flag "o" and the division-by-zero flag "z" are set indicates that the value represents a result of a computation that involved an overflow (this from the overflow flag "o"), as well as an attempt to divide by zero (this from the division-by-zero flag "z"). In one embodiment, the flags may provide the same status information as provided by, for example, information stored in a floating point status register (not shown) in a floating point unit that includes floating point status information testing circuit 10.

[061] In another embodiment, values in the other formats can be indicated as being inexact based in part on a particular bit, such as the least-significant bit f_{lsb} of their fraction fields. In that embodiment, that particular bit operates as an inexact flag. Thus, the value will be indicated as being inexact if that particular bit, such as the bit f_{lsb}, has the value "one." Otherwise, the operand has an exact status.

[062] With this background on the exemplary operand formats, the structure and operation of exemplary tester unit 10 will be described in connection with Figure 1. With reference to Figure 1, the exemplary tester unit 10 generally includes an operand buffer 11, an operand analysis circuit 12, and a result generator circuit 13. In basic operation, the operand buffer 11 receives and stores an operand from an external memory storage device, such as a set of general memory registers, in a conventional manner. The operand analysis circuit 12 analyzes the operand in the operand buffer 11 and generates signals providing information relating to the operand. These signals are provided to the result generator circuit 13. The signals provided by the operand analysis circuit 12 essentially provide information as to the type or format of the respective operand, such as indicating whether the operand is in the zero format 100, underflow format 101, de-normalized format 102, normalized non-zero format 103, the overflow format 104, infinity format 105, or the NaN format 106.

[063] The result generator circuit 13 receives the signals from the operand analysis circuit 12, control signals A through L, and a signal from operand buffer 11. In response, the result generator circuit 13 generates a result signal. In an embodiment of the present invention, the control signals are provided by a control

unit (not shown) within the floating point unit that includes exemplary tester unit 10. In more detail, these control signals may indicate the particular items of floating point status information that are to be tested in relation to the format of the operand within operand buffer 11. The result generator circuit 13 generates a result signal indicating whether the operand in operand buffer 11 conforms to the state specified by the control signals. Additionally, it will be appreciated that the result signal may be directly or indirectly coupled to the control unit for controlling its subsequent operations, *e.g.*, an outcome of a conditional floating point operation or instruction.

[064] Before proceeding to a detailed description of exemplary tester unit 10, it would be helpful to further describe the operation of exemplary tester unit 10 in relation to the control signals consistent with an embodiment of the present invention. In the illustrated embodiment, exemplary tester unit 10 receives twelve control signals, *i.e.*, signals A-L. Control signals A and B control the operation of tester unit 10 in relation to control signals C-K. The L control signal enables the result indicated by the result signal to be complemented. Using controls signals A, B, and L, exemplary tester unit 10 selectively processes the operand to determine the appropriate value of the result signal.

[065] In more detail regarding the exemplary embodiment, if both control signals A and B are negated and the L control is negated, asserting following additional control signals will result in the following values of the result signal:

[066] (a) If the C control signal is asserted, the result signal is asserted if the operand in operand buffer 11 is in the NaN format 106 or the infinity

format 105 and the "invalid operation" status flag ("n") is set, in which case the bit f_{lsb+4} of the fraction field has the value "one."

[067] (b) If the D control signal is asserted, the result signal is asserted if the operand in operand buffer 11 is in the NaN format 106 or the infinity format 105 and the "overflow" status flag ("o") is set, in which case the bit f_{lsb+3} of the fraction field has the value "one."

[068] (c) If the E control signal is asserted, the result signal is asserted if the operand in operand buffer 11 is in the NaN format 106 or the infinity format 105 and the "underflow" status flag ("u") is set, in which case the bit f_{lsb+2} of the fraction field has the value "one."

[069] (d) If the F control signal is asserted, the result signal is asserted if the operand in operand buffer 11 is in the NaN format 106 or the infinity format 105 and the "zero" status flag ("z") is set, in which case the bit f_{lsb+1} of the fraction field has the value "one."

[070] (e) If the G control signal is asserted, the result signal is asserted if the operand in operand buffer 11 is in the NaN format 106 or the infinity format 105 and the "inexact" status flag ("x") is set, in which case the bit f_{lsb} of the fraction field has the value "one."

[071] (f) If the H control signal is asserted, the result signal is asserted if the operand in operand buffer 11 is in the overflow format 104.

[072] (g) If the I control signal is asserted, the result signal is asserted if the operand in operand buffer 11 is in the underflow format 101.

[073] (h) If the J control signal is negated and the K control signal is asserted, the result signal is asserted if the operand in operand buffer 11 is in the de-normalized format 102.

[074] (i) If the J Control signal is asserted and the K control signal is negated, the result signal is asserted if the operand in operand buffer 11 is in the de-normalized format 102 or normalized non-zero format 103 and the inexact flag is set, in which case the least-significant bit f_{lsb} of the fraction field has the value "one."

[075] (j) If both the J control signal and the K control signal are asserted, the result signal is asserted if the operand in operand buffer 11 is in the de-normalized format 102 and the inexact flag is set, in which case the least-significant bit f_{lsb} of the field has the value "one."

[076] Alternatively, if both control signals A and B are asserted and the L control is negated, asserting following additional controls signal will result in the following values of the result signal:

[077] (k) If the C control signal is asserted, the result signal is asserted if the operand in operand buffer 11 is in the NaN format 106.

[078] (l) If the D control signal is asserted, the result signal is asserted if the operand in operand buffer 11 is in the infinity format 105.

[079] (m) If the E control signal is asserted, the result signal is asserted if the operand in operand buffer 11 is in the overflow format 104.

[080] (n) If the F control signal is asserted, the result signal is asserted if the operand in operand buffer 11 is in the normalized non-zero format

103 with the inexact flag "x" clear, in which case the bit f_{lsb} of the fraction field has the value "zero."

[081] (o) If the G control signal is asserted, the result signal is asserted if the operand in operand buffer 11 is in the normalized non-zero format 103 with the exact flag "x" set, in which case the bit f_{lsb} of the fraction field has the value "one."

[082] (p) If the H control signal is asserted, the result signal is asserted if the operand in operand buffer 11 is in the de-normalized format 102 with the inexact flag "x" clear, in which case the bit f_{lsb} of the fraction field has the value "zero."

[083] (q) If the I control signal is asserted, the result signal is asserted if the operand in operand buffer 11 is in the de-normalized format 102 with the inexact flag "x" set, in which case the bit f_{lsb} of the fraction field has the value "one.

[084] (r) If the J control signal is asserted, the result signal is asserted if the operand in operand buffer 11 is in the underflow format 101.

[085] (s) If the K control signal is asserted, the result signal is asserted if the operand in operand buffer 11 is in the zero format 100.

[086] If the A control signal is negated and the B control signal is asserted, the exemplary tester unit 10 operates as described above in connection with (k) through (s) except that the result signal will be asserted only if the operand in operand buffer 11 is negative. Further, if the A control signal is asserted and the B control signal is negated the exemplary tester unit 10 operates as described above

in connection with (k) through (s) except that the result signal will be asserted only if the operand in operand buffer 11 is positive. If the L signal is asserted, the result signal will be the complement of that described above.

[087] With this background on the exemplary embodiment and with reference to Figure 1, the exemplary operand analysis circuit 12 is a circuit having logical circuitry, such as comparators and other logical gates, that serve to identify the format of the operand in operand buffer 11. For example, exemplary operand analysis circuit 12 includes comparators 20, 21, 22, 30, 31, 32, 33, and 34 that respond to the contents of operand buffer 11.

[088] In the exemplary embodiment, comparator 20 generates an asserted signal if the bits $e_{msb}\cdots e_{lsb}$ of the exponent field of the operand in operand buffer 11 are all binary one's, which will be the case if the operand is in the infinity format 105 or the NAN format 106.

[089] Comparator 21 generates an asserted signal if the bits $e_{msb}\cdots e_{lsb+1}$ of the exponent field of the operand in the operand buffer 11 are all binary one's, and the bit e_{lsb} is a binary zero, which will be the case if the operand is in the overflow format 104.

[090] Comparator 22 generates an asserted signal if the bits $e_{msb}\cdots e_{lsb}$ of the exponent field of the operand in operand buffer 11 are all binary zero's, which will be the case if the operand is in the zero format 100, underflow format 101, or de-normalized format 102.

[091] Comparator 30 generates an asserted signal if the bits $f_{msb}\cdots f_{lsb+5}$ of the fraction field of the operand in the operand buffer 11 are all binary one's, which

may be the case if the operand is in the de-normalized format 102, normalized non-zero format 103, overflow format 104, or NaN format 106.

[092] Comparator 31 generates an asserted signal if the bits $f_{msb}\cdots f_{lsb+5}$ of the fraction field of the operand in the operand buffer 11 are all binary zero's, which may be the case if the operand is in the zero format 100, underflow format 102, de-normalized format 102, normalized non-zero format 103 or infinity format 105.

[093] Comparator 32 generates an asserted signal if the bits $f_{lsb+4}\cdots f_{lsb}$ of the fraction field of the operand in the operand buffer 11 are all binary one's, which may be the case if the operand is in the de-normalized format 102 or normalized non-zero format 103 and which will be the case in the overflow format 104, or if all of the flags "n," "o," "u," "z," and "x" are set in the infinity format 105 or NaN format 106

[094] Comparator 33 generates an asserted signal if the bits $f_{lsb+4}\cdots f_{lsb+1}$ of the fraction field of the operand in the operand buffer 11 are binary zero's and the bit f_{lsb} of the fraction field is a binary "one," which will be the case if the operand is in the underflow format 101 and which may be the case if the operand is in the de-normalized format 102, normalized non-zero format 103, or if the flags "n," "o," "u," and "z" are clear and the flag "x" is set in the infinity 105 or NaN format 106.

[095] Comparator 34 generates an asserted signal if all of the bits $f_{lsb+4}\cdots f_{lsb+1}$ of the fraction field of the operand in the operand buffer 11 are binary zero's, which will be the case if the operand is in the zero format 100, and which may be the case if the operand is in the denormalized format 102, normalized non-zero format 103, or if the flags "n," "o," "u," "z" and "x" are clear in the infinity format 105 or NaN format 106.

[096] Gates 35-42 respond to the output of the above mentioned comparators to identify the status of the operand in operand buffer 11. In the exemplary embodiment, AND gate 35 will generate an asserted signal if comparators 31 and 34 are both generating asserted signals. This will be the case if the bits $f_{msb}\cdots f_{lsb}$ of the fraction field of the operand in the operand buffer 11 have the bit pattern 00000000000000000000000.

[097] AND gate 36 will generate an asserted signal if comparators 22, 31 and 33 are all generating asserted signals. This will be the case if the bits $e_{msb}\cdots e_{lsb}$ of the fraction field of the operand in the operand buffer 11 have the bit pattern 00000000 and the bits $f_{msb}\cdots f_{lsb}$ of the fraction field of the operand in the operand buffer 11 have the bit pattern 00000000000000000000001, in which case the operand is in the underflow format 101.

[098] AND gate 37 will generate an asserted signal if comparators 21, 30 and 32 are all generating asserted signals. This will be the case if the bits $e_{msb}\cdots e_{lsb}$ of the fraction field of the operand in the operand buffer 11 have the bit pattern 11111110 and the bits $f_{msb}\cdots f_{lsb}$ of the fraction field of the operand in the operand buffer 11 have the bit pattern 11111111111111111111111, in which case the operand is in the overflow format 104.

[099] NAND gate 38 will generate an asserted signal if the comparator 20 is generating an asserted signal and the comparator 31 is generating a negated signal. This will be the case if the bits $e_{msb}\cdots e_{lsb}$ of the fraction field of the operand in the operand buffer 11 have the bit pattern 11111111 and at least one of bits

$f_{msb}\cdots f_{lsb+4}$ of the fraction field of the operand in the operand buffer 11 has the value "one," in which case the operand is in the NaN format 106.

[0100] AND gate 39 will generate an asserted signal if the comparator 20 is generating an asserted signal and the comparator 31 is generating an asserted signal. This will be the case if the bits $e_{msb}\cdots e_{lsb}$ of the fraction field of the operand in the operand buffer 11 have the bit pattern 11111111 and the bits $f_{msb}\cdots f_{lsb+4}$ of the fraction field of the operand in the operand buffer 11 have the bit pattern 00000000000000000 , in which case the operand is in the infinity format 105.

[0101] NAND gate 40 will generate an asserted signal if the comparator 22 is generating an asserted signal and AND gates 35 and 36 are generating negated signals. This will be the case if the bits $e_{msb}\cdots e_{lsb}$ of the fraction field of the operand in the operand buffer 11 have the bit pattern 00000000, at least one of the bits $f_{msb}\cdots f_{lsb}$ of the fraction field of the operand in the operand buffer 11 has the value "one," and the operand is not in the underflow format 101, in which case the operand is in the denormalized format 102.

[0102] AND gate 41 will generate an asserted signal if the comparator 22 is generating an asserted signal and AND gate 35 is generating an asserted signal. This will be the case if the bits $e_{msb}\cdots e_{lsb}$ of the fraction field of the operand in the operand buffer 11 have the bit pattern 00000000 and the bits $f_{msb}\cdots f_{lsb}$ of the fraction field of the operand in the operand buffer 11 have the bit pattern 00000000000000000000000, in which case the operand is in the zero format 100.

[0103] Finally, NAND gate 42 will generate an asserted signal if the AND gate 37 and comparators 20 and 22 are all generating negated signals. This will be the case if the operand is in the normalized non-zero format 103.

[0104] In the exemplary embodiment, the result generator circuit 13 essentially includes four sections, including an A/B signal decoder section 50, an AB negated section 51, an AB non-negated section 52 and a combiner section 53. The AB negated section 51 receives the signals from the operand analysis circuit 12 and control signals C-K and generates a signal (referred to as the AB NEG signal) out of AND gate 91 representative of a result as described above in connection with items (a) through (j). Similarly, the AB non-negated section 52 receives the signals from the operand analysis circuit 12 and control signals C-K and generates a signal (referred to as the AB NON-NEG signal) out of AND gate 92 representative of a result as described above in connection with items (k) through (s). The A/B signal decoder 50 receives control signals A-B and a signal representative of the sign bit of the operand in operand buffer 11 to generate gating signals. These gating signals enable the combiner section 53 to selectively gate one or none of the AB NEG signal or the AB NON-NEG signal as a preliminary result signal (*i.e.*, the signal generated by gate 93 and referred to as PRE). The combiner section 53 also receives the L control signal which controls the generation of the result signal (*i.e.*, the output of gate 94) as either the true or complement of the PRE signal.

[0105] Looking at the exemplary embodiment in more detail, the A/B signal decoder 50 includes NAND gate 60, AND gate 61, NAND gate 62 and OR gate 63. The NAND gate 60 receives the control signals A-B and generates an asserted

signal if both are negated. The AND gate 61 receives the B signal and a signal representative of the sign bit of the operand in operand buffer 11 and generates an asserted signal if both are asserted. It will be appreciated that if the signal representative of the sign bit is asserted, the operand will be negative. The NAND gate 62 receives control signal A and a signal representative of the sign bit of the operand in operand buffer 11 and generates an asserted signal if the A signal is asserted and the signal representative of the sign bit is negated, the operand will be positive. The OR gate 63 receives the signals from both AND gate 61 and NAND gate 62 and generates an asserted signal if either signal is asserted.

[0106] Accordingly, the OR gate 63 will generate an asserted signal if:

[0107] (A) the A and B control signals are both asserted, regardless of the sign of the operand in operand buffer 11,

[0108] (B) the B control signal is asserted and the operand in operand buffer 11 is negative, or

[0109] (C) the A control signal is asserted and the operand buffer 11 is positive.

[0110] The exemplary AB negated section 51 includes logic gates 70-80. In more detail, AND gate 70 (see reference item (a) above) generates an asserted signal if the signal from comparator 20, a signal representative of bit f_{lsb+4} of the fraction field of the operand in operand buffer 11, and the C control signal are all asserted. It will be appreciated that the AND gate 70 will generate an asserted signal if the operand in operand buffer 11 is in the infinity format 105 or NaN format 106, the invalid operation flag "n" is set and the C control signal is asserted.

[0111] AND gate 71 (see reference item (b) above) generates an asserted signal if the signal from comparator 20, a signal representative of bit f_{lsb+3} of the fraction field of the operand in operand buffer 11, and the D control signal are all asserted. It will be appreciated that the AND gate 71 will generate an asserted signal if the operand in operand buffer 11 is in the infinity format 105 or NaN format 106, the overflow flag "o" is set and the D control signal is asserted.

[0112] AND gate 72 (see reference item (c) above) generates an asserted signal if the signal from comparator 20, a signal representative of bit f_{lsb+2} of the fraction field of the operand in operand buffer 11, and the E control signal are all asserted. It will be appreciated that the AND gate 72 will generate an asserted signal if the operand in operand buffer 11 is in the infinity format 105 or NaN format 106, the underflow flag "u" is set and the E control signal is asserted.

[0113] AND gate 73 (see reference item (d) above) generates an asserted signal if the signal from comparator 20, a signal representative of bit f_{lsb+1} of the fraction field of the operand in operand buffer 11, and the F control signal are all asserted It will be appreciated that the AND gate 73 will generate an asserted signal if the operand in operand buffer 11 is in the infinity format 105 or NaN format 106, the divide-by-zero flag "z" is set and the F control signal is asserted.

[0114] AND gate 74 (see reference item (e) above) generates an asserted signal if the signal from comparator 20, a signal representative of bit f_{lsb} of the fraction field of the operand in operand buffer 11, and the G control signal are all asserted. It will be appreciated that the AND gate 74 will generate an asserted

signal if the operand in operand buffer 11 is in the infinity format 105 or NaN format 106, the inexact flag "x" is set and the G control signal is asserted.

[0115] AND gate 75 (see reference item (f) above) generates an asserted signal if the signal from AND gate 37 and the H control signal are both asserted, it will be appreciated that the AND gate 74 will generate an asserted signal if the operand in operand buffer 11 is in the overflow format 104 and the H control signal is asserted.

[0116] AND gate 76 (see reference item (g) above) generates an asserted signal if the signal from AND gate 36 and the I control signal are both asserted. It will be appreciated that the AND gate 74 will generate an asserted signal if the operand in operand buffer 11 is in the underflow format 101 and the I control signal is asserted.

[0117] NAND gate array 77 (see reference item (i) above) generates an asserted signal if the signal from comparator 20 is negated, the J control signal is asserted and the K control signal is negated, the signals from AND gates 36 and 37 are negated and a signal representative of bit f_{lsb} of the fraction field of the operand in operand buffer 11 is asserted. Furthermore, those skilled in the art will appreciate that the NAND gate array 77 will generate an asserted signal if:

[0118] (A) the J control signal is asserted and the K control signal is negated, AND

[0119] (B) the operand in operand buffer 11 in the denormalized format 102 or the normalized non-zero format 103, AND

[0120] (C) the inexact flag "x" is set.

[0121] NAND gate 78 (see reference item (h) above) generates an asserted signal if the signal from NAND gate 40 is asserted and if the J control signal is negated and the K control signal is asserted. It will be appreciated that the NAND gate 78 will generate an asserted signal if the operand in operand buffer 11 is the denormalized format 102 and the J control signal is negated and the K control signal is asserted.

[0122] NAND gate 79 (see reference item (j) above) generates an asserted signal if the signal from NAND gate 40 is asserted, if the signal representative of the bit f_{lsb} of the fraction field of the operand in operand buffer 11 is asserted, and the J and K control signals are both asserted. It will be appreciated and the NAND gate 78 will generate an asserted signal if:

[0123] (A) the J and K control signals are both asserted, AND

[0124] (B) the operand in operand buffer 11 is in the denormalized format 102, AND

[0125] (C) the inexact flag "x" is set.

(i) Finally, the AB negated section 51 also includes an OR network 80, which generates an asserted signal if the signal from any of AND or NAND gates 70 through 79 is asserted.

[0126] The exemplary AB non-negated section 52 includes logic gates 81-90. In more detail, AND gate 81 (see reference item (k) above) generates an asserted signal if the signal from NAND gate 38 and the C control signal are both asserted. This will be the case if the operand in operand buffer 11 is in the NaN format 106 and the C control signal is asserted.

[0127] AND gate 82 (see reference item (l) above) generates an asserted signal if the signal from AND gate 39 and the D control signal are both asserted. This will be the case if the operand in operand buffer 11 is in the infinity format 105 and the D control signal is asserted.

[0128] AND gate 83 (see reference item (m) above) generates an asserted signal if the signal from AND gate 37 and the E control signal are both asserted. This will be the case if the operand in operand buffer 11 is in the overflow format 104 and the E control signal is asserted.

[0129] NAND gate 84 (see reference item (n) above) generates an asserted signal if the signal from NAND gate 42 is asserted, a signal representative of the bit f_{lsb} of the operand in operand buffer 11 is negated and the F control signal is asserted. This will be the case if the operand in operand buffer 11 is in the normalized non-zero 103 with the inexact flag "x" clear and the F control signal is asserted.

[0130] AND gate 85 (see reference item (o) above) generates an asserted signal if the signal from AND gate 42 is asserted, a signal representative of the bit f_{lsb} of the operand in operand buffer 11 is asserted and the G control signal is asserted. This will be the case if the operand in operand buffer 11 is in the normalized non-zero format 103 with the inexact flag "x" set and the G control signal is asserted.

[0131] AND gate 86 (see reference item (p) above) generates an asserted signal if the NAND gate 40 is generating an asserted signal, a signal representative of the bit f_{lsb} of the operand in operand buffer 11 is negated and the H control signal

is asserted. This will be the case if the operand buffer 11 is in the denormalized format 102 with the inexact "x" clear and the H control signal asserted.

[0132] NAND gate 87 (see reference item (q) above) generates an asserted signal if the NAND gate 40 is generating an asserted signal, a signal representative of the bit f_{lsb} of the operand in operand buffer 11 is asserted and the I control signal is asserted. This will be the case if the operand in operand buffer 11 is in the denormalized format 102 with the inexact flag "x" set and the I control signal asserted.

[0133] AND gate 88 (see reference item (r) above) generates an asserted signal if the AND gate 36 is generating an asserted signal and the J control signal is asserted. This will be the case if the operand in operand buffer 11 is in the underflow format 101 with the J control signal asserted.

(ii) AND gate 89 (see reference item (s) above) generates an asserted signal if the AND gate 41 is generating an asserted signal and the K control signal is asserted. This will be the case if the operand in operand buffer 11 is in the zero format 100 with the K control signal asserted.

(iii) Finally, the AB non-negated section 52 includes OR network 90, which generates an asserted signal if the signal from any of AND or NAND gates 81 through 89 is asserted.

[0134] The combiner section 53 receives the signals from OR network 80, OR network 90 and the L control signal, as well as tagging signals generated by the NAND gate 60 and OR gate 63 of the A/B signal decoder 50. Based upon these received signals, combiner section 53 generates the result signal. If the NAND 60

generates an asserted signal, which will be the case if both A and B control signals are negated (reference items (a) through (j) above), an AND gate 91 will be enabled to gate the signal from the OR network 80 of the A/B negated section 51 to one input of an OR gate 93 as an A/B NEG RES partial result signal.

[0135] However, it will be appreciated that OR gate 63 generates an asserted signal if:

[0136] (A) both A and B control signals are asserted,

[0137] (B) the B control signal is asserted and a signal representative of the sign bit of the operand is also asserted (*i.e.*, the operand in operand buffer 11 is negative), or

[0138] (C) the A control signal is asserted and a signal representative of the sign bit of the operand is negated (*i.e.*, the operand in operand buffer 11 is positive),

[0139] (see reference items (k) through (s) above). If OR gate 63 generates the asserted signal, AND gate 92 will be enabled to gate the signal from the OR network 90 of the AB non-negated section 52 to the other input of OR gate 93 as an A OR B NON-NEG RES partial result signal. It will be appreciated that, since only one of NAND gate 60 or OR gate 63 will be generating an asserted signal at any point in time, at most one of A/B NEG RES and A or B NON-NEG RES will be asserted. On the other hand, it will also be appreciated that neither partial result signal may be asserted. If either the A/B NEG RES or the A OR B NON-NEG RES partial result signal is asserted, the OR gate 93 generates an asserted PRE preliminary result signal, which is coupled to an XOR gate 94. The XOR gate 94

also receives the L control signal and generates the result signal that corresponds to the PRE preliminary result signal if the L control signal is negated and complements the PRE preliminary result signal if the L control signal is asserted.

[0140] It will be appreciated that a plurality of the C through K control signals may be asserted at any point in time for a given operand. This advantageously enables the tester unit 10 to test for the existence of more than one status condition contemporaneously on the given operand. Thus, a control unit (not shown) may be able to collectively use results from tests for multiple status conditions as part of processing a floating point instruction or conditioning the outcome of a conditional floating point operation.

[0141] It will be appreciated that a system in accordance with the invention can be constructed in whole or in part from special purpose hardware or a general purpose computer system, or any combination thereof, any portion of which may be controlled by a suitable program. Any program may in whole or in part comprise part of or be stored on the system in a conventional manner, or it may in whole or in part be provided into the system over a network or other mechanism for transferring information in a conventional manner. In addition, it will be appreciated that the system may be operated and/or otherwise controlled by means of information provided by an operator using operator input elements (not shown) which may be connected directly to the system or which may transfer the information to the system over a network or other mechanism for transferring information in a conventional manner.

[0142] The foregoing description has been limited to a specific embodiment of this invention. It will be apparent, however, that various variations and modifications may be made to the invention, with the attainment of some or all of the advantages of the invention. It is the object of the appended claims to cover these and such other variations and modifications as come within the true spirit and scope of the invention.

[0143] Other embodiments of the invention will be apparent to those skilled in the art from consideration of the specification and practice of the invention disclosed herein. It is intended that the specification and examples be considered as exemplary only, with a true scope and spirit of the invention being indicated by the following claims.